Escalas

“Las escalas son funciones que mapean un dominio de datos de entrada a un rango de datos de salida.”

Definición de escalas, según Mike Bostock

No es usual que exista una correspondencia exacta entre los datos de un conjunto y las medidas de los pixeles que se van a utilizar para una visualización. Las escalas proveen una manera conveniente de mapear los datos originales a nuevos valores que son útiles para el propósito de visualización.

Las escalas de D3 son *funciones* en las que quién hace la visualización es quien define los parámetros. Una vez creada esta función, se invoca, se le pasa un valor y ésta devuelve el valor de salida, ya con el factor de conversión. Uno puede definir y usar tantas escalas como quiera.

Puede ser tentador pensar en una escala como algo que aparece visualmente en la imagen final – como una serie de marcadores que muestran una progresión de datos. *No se deje engañar!* Estos marcadores son parte del *eje*, el cual es en esencia la representación visual de la escala. Una escala es una relación matemática, que no tiene un resultado visual como tal. Le recomiendo que piense en escalas y ejes como dos elementos que están relacionados.

Este tema describe únicamente escalas *lineales* puesto que son las más comunes y fáciles de entender. Una vez entienda las escalas lineales, entender las demás va a ser muy fácil.

Manzanas y Pixeles

Imagínese que el siguiente conjunto de datos representa el número de manzanas que se vende mensualmente en una tienda de frutas a lo largo de la carretera.

```
var dataset = [ 100, 200, 300, 400, 500 ];
```

Primero que todo, esto es buena noticia, pues la tienda está vendiendo 100 manzanas adicionales cada mes! El negocio está creciendo. Para mostrar el éxito, usted quiere mostrar en un diagrama de barras la curva empinada hacia arriba desplegando las ventas de manzanas, en la que cada dato corresponde a la altura de una barra.

Hasta el momento, hemos usado los datos directamente para su despliegue y se ha

ignorado la diferencia de unidades. Entonces, si se vendieron 500 manzana, esto significa que la barra tendría un tamaño de 500 pixeles.

Eso puede que funcione, pero que tal si al mes siguiente se venden 600 manzanas? Y que el año que viene se venden 1,800 manzanas? Las personas que ven sus datos tendrían que adquirir monitores más grandes solamente para ver el tamaño real de esas barras de manzanas! (Mmm, barras de manzanas!)

Acá es donde entran a jugar las escalas. Puesto que las manzanas no son pixeles (tampoco son naranjas), necesitamos utilizar escalas para traducir de una a la otra.

Dominios y Rangos

El *dominio de entrada* de una escala es el rango de los valores iniciales. Con los datos de manzanas de arriba, el dominio de entrada sería 100 y 500 (para el mínimo y el máximo) o cero y 500.

El *rango de salida* de la escala es el rango de valores posibles de salida, que usualmente se denominan valores de despliegue y usan pixeles por unidad de medida. El rango de salida lo decide usted como diseñador. Si usted decide que la barra más pequeña tendrá una altura de 10 pixeles y que la más alta tendrá 350 pixeles, entonces usted puede definir el rango de salida de 10 a 350.

Por ejemplo, puede crear una escala donde el dominio de entrada es `100, 500` y el rango de salida es de `10, 350`. Si usted le da a esa escala un valor de `100`, ésta le devolverá `10`.

Si usted le da `500`, le devolverá `350`. Si le da `300`, le entrega `180` en una bandeja de plata. (`300` es el centro del dominio y `180`es el centro del rango.)

Podemos visualizar el dominio y el rango como dos ejes correspondientes, uno junto al otro:

Dominio de Entrada 100 300 500 10 180 350 Rango de Salida
Un cosa más. Dado que es muy fácil mezclar la terminología de *dominio de entrada* y *rango de salida*, les sugiero un pequeño ejercicio. Cuando diga “entrada”, usted dice “dominio”. Luego digo “salida” y usted dice “rango.

Listo? Bueno:

- Entrada! Dominio!
- Salida! Rango!
- Entrada! Dominio!
- Salida! Rango!

Entendió? Muy bien.

Normalización

Si está familiarizado con el concepto de *normalización*, puede ser conveniente que sepa que eso es todo lo que está pasando aquí con una escala lineal.

La normalización es el proceso de mapeo de un valor numérico a un nuevo valor que está entre 0 y 1, basado en los posibles valores mínimos y máximos. Por ejemplo, con 365 días al año, el día número 310 se mapea a más o menos 0.85 o 85% del recorrido del año.

Con escalas lineales, dejamos que D3 se encargue de la matemática que está detrás del proceso de normalización. El valor de entrada se normaliza de acuerdo con el dominio y el valor normalizado se cambia de escala para el rango de salida.

Creación de una Escala

Los generadores de escala de D3 se puede acceder con `d3.scale` seguidos por el tipo de escala que se desee.

```
var scale = d3.scale.linear();
```

Felicitaciones! Ya `scale` es una función a la que le puede pasar datos de entrada. (No se deje confundir con `var` arriba; recuerde que en JavaScript las variables pueden guardar funciones.)

```
scale(2.5); //Devuelve 2.5
```

Puesto que no hemos definido un dominio ni un rango, esta función está mapeando la entrada a la salida con una escala de 1:1. Esto quiere decir que lo que ingrese saldrá sin ningún cambio.

Podemos fijar el dominio de entrada de la escala en `100, 500` si le pasamos esos valores como un arreglo al método `domain()`.

```
scale.domain([100, 500]);
```

y los del rango de salida en forma similar, con `range()`:

```
scale.range([10, 350]);
```

Estos dos pasos se pueden hacer por separado, como se acaba de mostrar, o se pueden encadenar en una línea de código:

```
var scale = d3.scale.linear()
              .domain([100, 500])
              .range([10, 350]);
```

En cualquier caso, nuestra escala está lista!

```
scale(100);  //Devuelve 10
scale(300);  //Devuelve 180
scale(500);  //Devuelve 350
```

Por lo general, las funciones de escala se llaman desde el método `attr()` u otros similares. Cambiemos la visualización del Diagrama de Dispersión de tal manera que use escalas dinámicas.

Definiendo la Escala de un Diagrama de Dispersión

Revisemos los datos del Diagrama de Dispersión:

```
var dataset = [
            [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
            [410, 12], [475, 44], [25, 67], [85, 21], [220, 88]
          ];
```

Si recuerda, `dataset` es un arreglo de arreglos. Mapeamos el primer valor en cada arreglo al eje x y el segundo valor al eje y. Empecemos con el eje x.

Mirando por encima los valores de x, vemos que el rango va de 5 a 480, por consiguiente un dominio de entrada razonable puede ser `0, 500`, cierto?

...

Por qué me mira de esa manera? Ah, porque quiere que su código siga siendo flexible y

escalable, de tal manera que funcione cuando cambien los datos en el futuro. Muy inteligente!

En vez de especificar valores fijos para el dominio, podemos usar funciones que se aplican a los arreglos y que son muy convenientes, tales como `min()` y `max()` para analizar el conjunto de datos sobre la marcha. Por ejemplo, este código pasa por todos los valores de x en nuestros arreglos y devuelve el valor del mayor:

```
d3.max(dataset, function(d) {    //Devuelve 480
    return d[0];  //Referencia el primer valor del sub-arreglo
});
```

Si se junta todo, podemos crear una función para cambiar la escala de nuestro eje de x:

```
var xScale = d3.scale.linear()
                .domain([0, d3.max(dataset, function(d) { return d[0]; })])
                .range([0, w]);
```

Primero, observe que la nombramos `xScale.` Por supuesto que se puede usar cualquier nombra para la escala, pero uno como `xScale` ayuda a recordarme lo que hace esta función.

Segundo, tome nota de que le asigné cero al rango inferior del dominio de entrada. (De otra manera, habría podido usar `min()` para calcular un valor dinámico.) El punto superior del dominio se fijó en el valor máximo del arreglo `dataset` (que es 480).

Por último, observe que el rango de salida quedó en `0` y `w`, el ancho del SVG.

Vamos a usar un código similar para crear la función de escala en el eje y.

```
var yScale = d3.scale.linear()
                .domain([0, d3.max(dataset, function(d) { return d[1]; })])
                .range([0, h]);
```

Observe que la función `max()` referencia a `d[1]`, el valor y de cada sub-arreglo. También, el punto superior del `range()` está definido como `h` en vez de `w`.

Las funciones de cambio de escala están en su sitio! Ahora lo único que falta es usarlas. Simplemente toca modificar el código donde se crea un `circle` (círculo) para cada dato.

```
.attr("cx", function(d) {
    return d[0];
    })
```

Para devolver un valor ajustado a la escala (en vez del valor original):

```
.attr("cx", function(d) {
    return xScale(d[0]);
    })
```

Igualmente, para el eje y, éste

```
.attr("cy", function(d) {
    return d[1];
    })
```

se cambia a:

```
.attr("cy", function(d) {
    return yScale(d[1]);
    })
```

Y por añadidura, hagamos el mismo cambio con las coordenadas de las etiquetas, entonces estas líneas

```
.attr("x", function(d) {
    return d[0];
    })
    .attr("y", function(d) {
        return d[1];
        })
```

se convierten en:

```
.attr("x", function(d) {
    return xScale(d[0]);
})
.attr("y", function(d) {
    return yScale(d[1]);
    })
```

Y ya está!

Acá encuentra el código funcionando. Visualmente, es tan decepcionante como el Diagrama de Dispersión original! Sin embargo, hemos progresado más de los que parece.

Refinando el Diagrama de Dispersión

Seguramente ha notado que los valores *y* más pequeños están ubicados en la parte alta de la gráfica, y que los más grandes están en la parte baja. Ya que estamos usando escalas, es muy fácil cambiar esto, de tal manera que los mayores valores se muestren en la parte superior, como es de esperarse. Es solo cuestión de cambiar el rango de salida de `yScale` de

```
.range([0,h]);
```

a

```
.range([h,0);
```

Acá encuentra este código. Sí, ahora una *menor* entrada a `yScale` produce un valor de salida *mayor*, de tal forma que los `circle`(círculos) y el `text` (texto) se empujan hacia abajo, más cerca de la base de la imagen. Yo sé, parece demasiado fácil!

Sin embargo, algunos de los elementos salen cortados. Debemos introducir una variable de `padding`.

```
var padding = 20;
```

Acá entonces podemos incorporar la cantidad de `padding` cuando estemos definiendo el rango de las dos escalas. El rango de `xScale` era `range([0, w])`, y ahora queda como

```
.range([padding, w - padding])
```

El rango de `yScale` era `range([h, 0])`, y ahora es

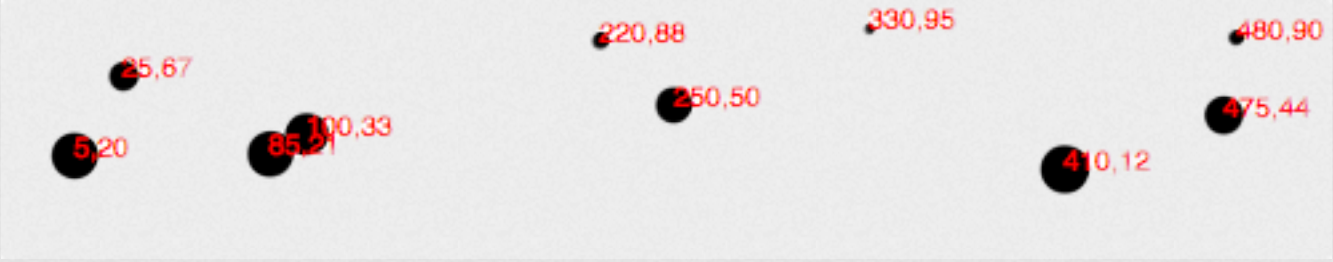
```
.range([h - padding, padding]);
```

Esto nos debe dar 20 pixeles adicionales de espacio en los bordes izquierdo, derecho,

superior e inferior del SVG. Y efectivamente así es!

Pero las etiquetas en el costado derecho todavía están cortadas, entonces toca duplicar el tamaño del relleno (padding) de `xScale`, entonces se multiplica por dos:

```
.range([padding, w - padding * 2]);
```



Mejor! Acá está el código. Pero aún toca hacer un cambio. En vez de definir el radio de cada `circle` (círculo) como la raíz cuadrada de su valor y (que es un “hack” y realmente tampoco es muy útil), por qué no crear una escala a la medida?

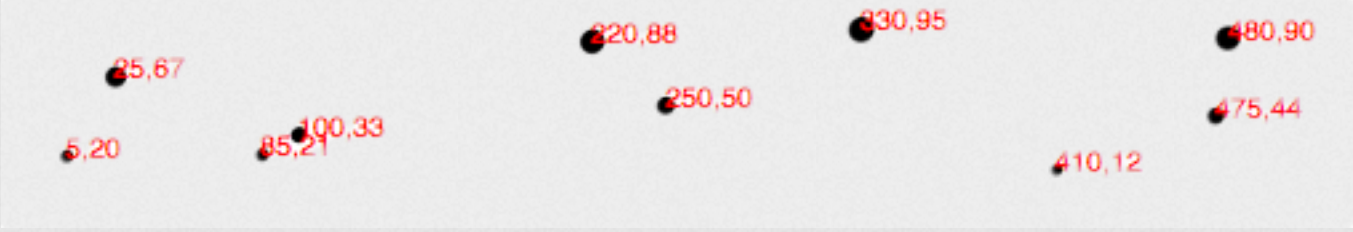
```
var rScale = d3.scale.linear()
                .domain([0, d3.max(dataset, function(d) { return d[1]; })])
```

```
            .range([2, 5]);
```

Así, para definir el valor del radio, se programa de esta manera:

```
.attr("r", function(d) {
    return rScale(d[1]);
    });
```

Esto es emocionante, porque de esta manera se garantiza que todos los valores de los radios *siempre* aparezcan dentro del rango `2,5`. (O * casi* siempre. Vea la referencia a `clamp()` debajo.) Los valores de `0` (el dato de entrada mínimo) recibirán círculos con radio de `2` (o un diámetro de 4 pixeles). Los valores más grandes se mostrarán como un círculo con radio de `5` (diámetro de 10 pixeles).

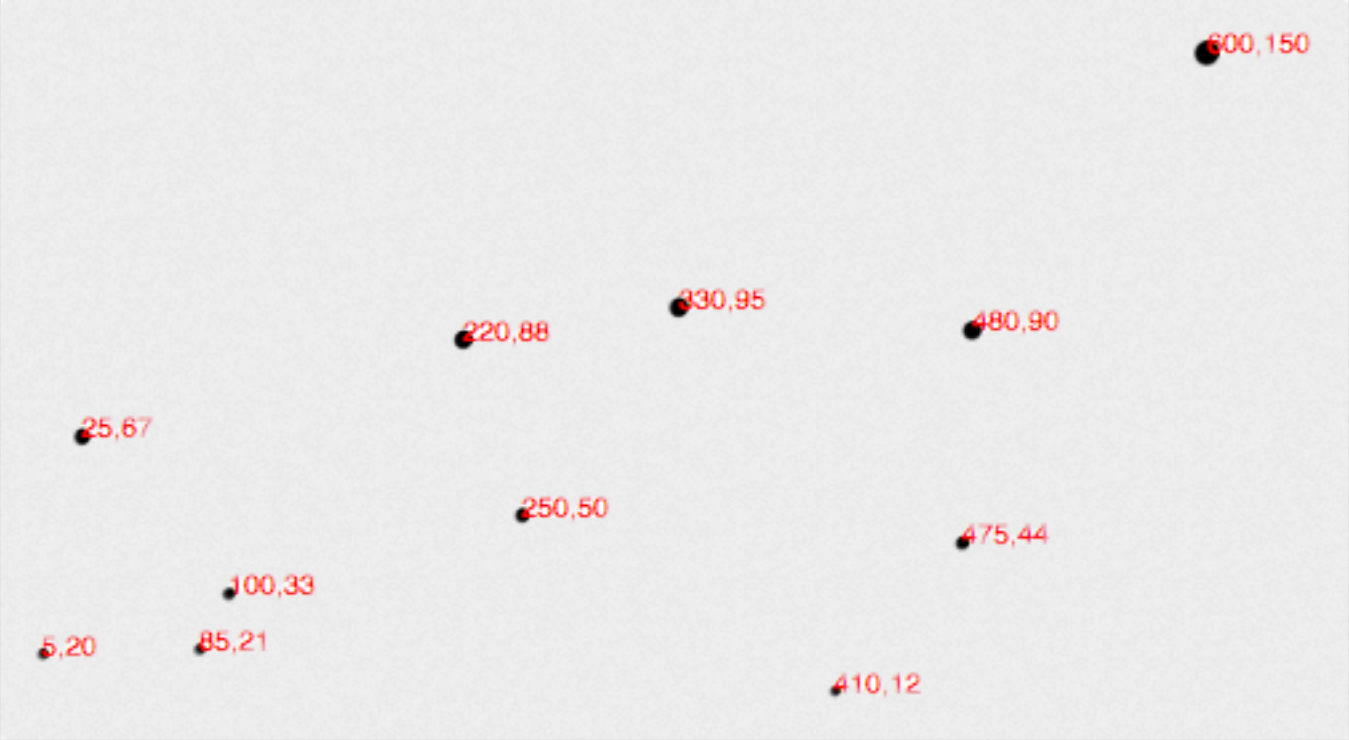


Voilá: Nuestra primera escala para una propiedad visual con un uso diferente al valor de un eje.

Por último y en el caso de que el poder de las escalas aún no lo haya descrestado, me gustaría añadir un arreglo al conjunto de datos: `[600, 150]`

Boom! Acá está el código. Puede notar cómo todos los puntos anteriores mantuvieron sus posiciones relativas, pero ahora están más cerca de sí mismos, más abajo y hacia la izquierda para acomodar al nuevo miembro.

Y ya, para una última revelación: Muy fácilmente podemos cambiar el tamaño de nuestro SVG, y todo cambiará de escala conforme al cambio. Acá he incrementado el valor de `h1` de `100` a `300` sin hacer *ningún otro cambio*:



Boom, nuevamente! El código actualizado. Ojalá que al ver esto pueda caer en cuenta que no es necesario pasar largas noches cambiando código porque un cliente decidió que la gráfica debería ser de 800 pixeles de ancho en vez de 600. Sí, va a poder dormir más gracias a mí (y los excelentes métodos que se hacen parte de D3). Estar descansado es una ventaja competitiva. Me dará las gracias próximamente.

Otros Métodos

`d3.scale.linear()` tiene otros métodos útiles que cabe mencionar brevemente acá:

- `nice()`– Esto le dice a la escala que tome cualquier dominio de entrada que se le haya pasado a `range()`, y lo expanda al valor redondeado más cercano. Del wiki de D3

> “Por ejemplo, del dominio [0.20147987687960267, 0.996679553296417], el dominio `nice`es [0.2, 1].” Esto es bastante útil para personas normales que usualmente encuentran algo difícil leer números como 0.20147987687960267.

`rangeRound()`– Use `rangeRound()` en lugar de `range()` y todos los valores a los que se le aplica la escala serán redondeados al número entero más cercano. Esto es útil si quiere que las figuras tengan valores exactos en pixeles, para que los bordes no se vean borrosos por causa de “antialiasing”.

`clamp()`– Por defecto, una escala lineal *puede* devolver valores que estén por fuera del rango especificado. Por ejemplo, si se da un valor por fuera del dominio de entrada esperado, la escala producirá un número que también está por fuera del rango de salida. Al llamar a `.clamp(true)` dentro de la escala, se encarga de que todos los valores queden dentro del rango especificado. Lo que esto significa es que los valores excesivos se redondearán a los valores bajos y altos del rango (aquel esté más cercano).

Otras Escalas

Adicionalmente a las escalas `lineales`(que se han explicado arriba), D3 incluye otros métodos para cambios de escala.

- `identity`– Una escala de 1:1 que se usa principalmente con valores en pixeles.
- `sqrt`– Una escala de raíz cuadrada
- `pow`– Una escala de potencia (buena para el gimnasio)

- `log` – Una escala logarítmica
- `quatize` – Una escala lineal con valores discretos para el rango de salida, para aquellos casos en que quiere organizar datos en “buckets”.
- `ordinal` – Escalas ordinales utilizan valores que nos son cuantitativos (como nombres de categorías) como salida: perfecto para comparar peras y manzanas.

Siguiente: Ejes –>

Este tutorial fue traducido por Gabriel Coch
Todo el contenido fue desarrollado por y le pertenece a Scott Murray